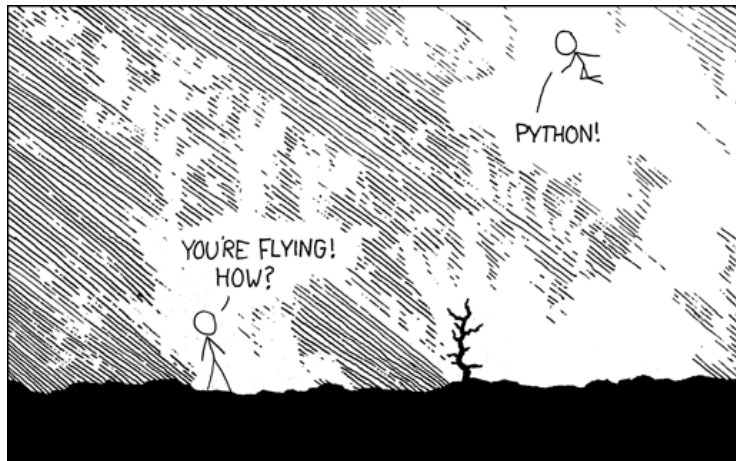




Introduction & setup

Practical Programming
in Chemistry





Not your typical course.

But a course, many chemists and I, would have loved to have during their Bachelor.

This course is all about how to talk (or write) to and use your computer...

So, it gets things done in chemistry, more efficiently.

Whether you go into computational chemistry, or not, you will produce data, and programming skills make a difference.

This is not what your friends in computer science learn.

It is much more practical.

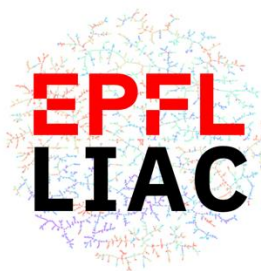
Note: You will most likely learn more during the exercises and project, than the lecture.

Grew up in Fribourg, Switzerland
- French
- Swiss German / German



NCCR
Catalysis

TT Assistant Professor
in Digital Chemistry
since Feb 2022



EPFL



Closing the loop...

Materials Science &
Engineering

Virtual screening &
simulation workflows
Prof Nicola Marzari

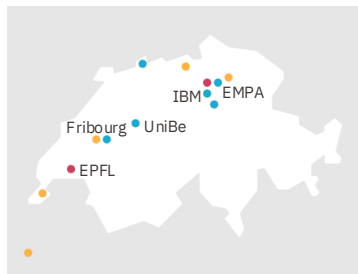
BSc ('14)/MSc ('16)



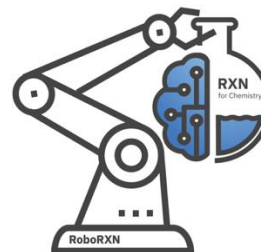
Empa

Materials Science and Technology

Lab work on ternary polymer
blends for organic solar cells
Prof Frank Nüesch



MPhil in Physics ('19)
Dr Alpha Lee



Machine learning for chemical synthesis
Intern/PhD/Postdoc
Dr Teodoro Laino

u^b

b
**UNIVERSITÄT
BERN**

PhD in Chemistry
and Molecular Sciences ('21)
Prof Jean-Louis Reymond

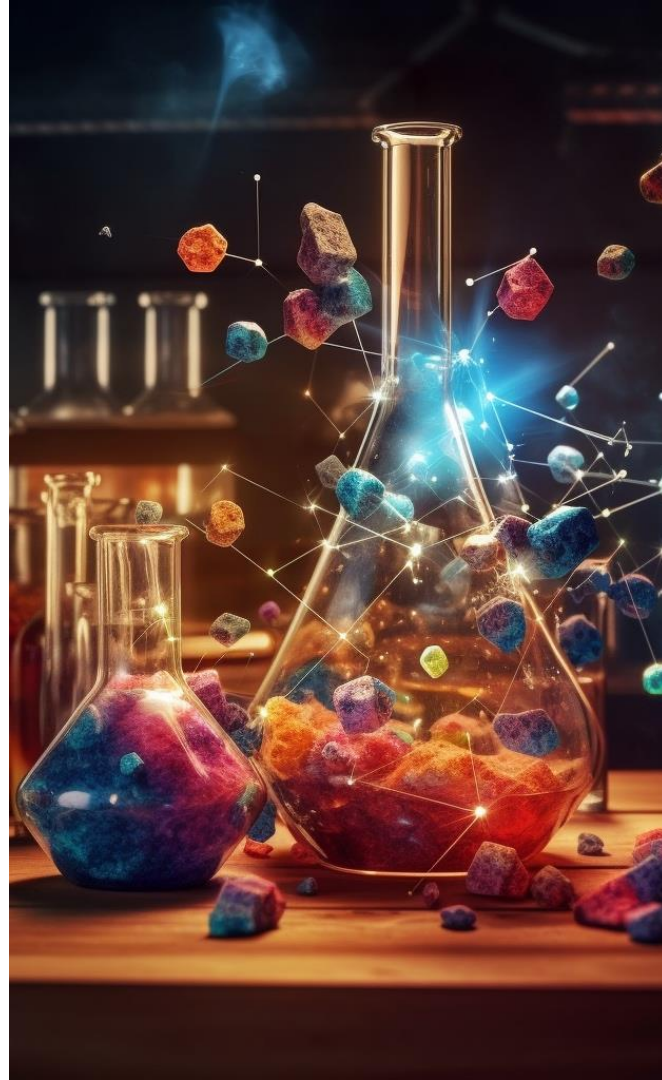


**UNIVERSITY OF
CAMBRIDGE**



IBM Research Europe

Mission: **Accelerating Discovery with Digital Chemistry**



PhD students

Bojana Ranković

Oliver Schilter (IBM Research)

Andres CM Bran

Junwu Chen

Jeff Guo

Victor Sabanza Gil (Luterbacher)

Paulo Neves (Janssen)

Rebecca Neeser (Correia, VantAI)

Sarina Kopf (Nevado)

Daniel Armstrong

Joshua Sin (Roche)

Sandro Agostini (IBM Research)

Théo Neukomm (Intel/Merck)

Salomé Guilbert (Röthlisberger)

Matt Hart (Trospha)



Rebecca



Sarina



Daniel

Joseph W. Abbott
jwa7

Admin

Annick Delmonaco

Postdocs/Engineers

Zlatko Jončev

Julian Götz

Edvin Fako

Sergey Pozdnyakov

Jeremy Goumaz

Project students

Shai Pranesh

Antonio Donez

Octavian Susanu

Arne Beckmann

Xuan Vu Nguyen

David Segura

Funding:



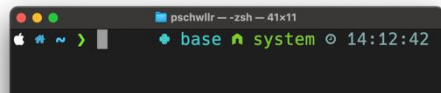
IBM Research



>15 nationalities — one team!
<https://schwallergroup.github.io>

EPFL The goal for this week – setup your computer

10



Programming language to communicate with computer system.
Super useful to know a few commands.

Basic command line usage
in **bash**



A better text editor
to handle Python code and everything
around it.

Install VS Code as interactive
development environment

General purpose programming language.
Create scripts, programs, webapps, ...

ANACONDA®

Install Python through anaconda
(main course programming language)

A platform to share, track,
and store code.

GitHub

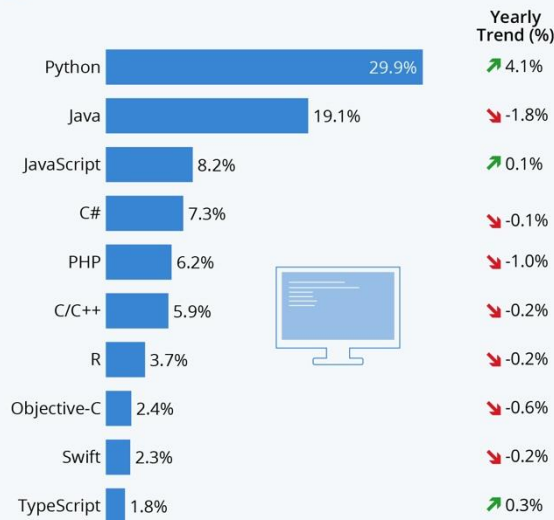
Creating a GitHub account
and getting familiar with basic commands

Main course programming language → Python

- Builds up on what you've learned last year
- Almost like writing/reading in English
- Versatile (data analysis, webpages, automation, visualisation, ...)
- Growing popularity in science communities
- Good for both simple and advanced scripting and applications

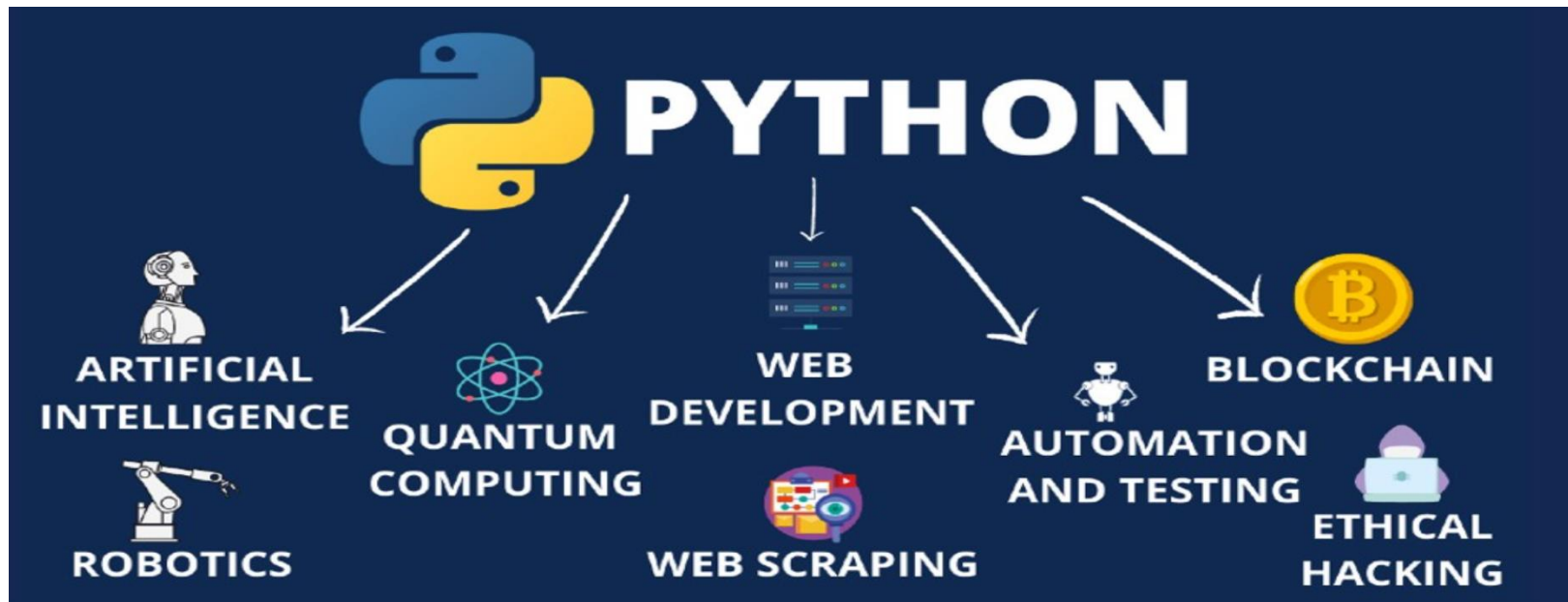
Python Remains Most Popular Programming Language

Popularity of each programming language based on share of tutorial searches in Google



Yearly trend compares percent change from Feb 2019 to Feb 2020
Sources: GitHub, Google Trends





<https://www.digiprima.com/files/wytrvqctb8ziq4e9icw2/python%20blog.jpg>

**Almost everything you can imagine
is just a pip/conda install away.**

Note: *pip install* is one of the commands you can use to install more functionality.

Compared to BA2 course

- Not only Python but the surrounding ecosystem (today you'll learn about bash)
- Focus on examples in chemistry (RDKit and other chemistry modules)
- Not only Python scripting, but how to make packages/small applications/...
- How to document and test
- How to use GitHub
- Write in Markdown
- Develop small applications to show your work to others (!!)
- Make Python packages that can be reused (!!)

- Slides on moodle
- Exercises: <https://github.com/schwallergroup/practical-programming-in-chemistry-exercises>
- Bring your own laptop!

Plan for this year

- Two courses per week for 6 weeks, 1 course in week 7, and the final project presentation in week 14.
- CH-200 Practical programming in Chemistry (total 14 lectures)
 - Wednesdays 10h-13h: 19. February - 2. April + 28. May in MED 0 1418
 - Thursdays 14h-17h: 20. February - 27. March in MEB 331
 - Office hours for the remaining of the semester 13h-14h in CH J2
- CH-343 Spectroscopy (total 13 lectures)
 - Wednesdays 10h-13h: 9. April - 21. May
 - Thursdays 14h-17h: 3. April - 22. May

Lecture	Topic	Exercise Link
01	Setup your environment	Lecture01
02	GitHub and creating first repositories	Lecture02
03	Conda, Jupyter notebooks, and Python basics	Lecture03
04	Advanced Python: file I/O, functions, error handling, and classes.	Lecture04
05	Numerical operations, data handling, data visualization: <code>numpy</code> , <code>pandas</code> , <code>matplotlib</code>	Lecture05
06	<code>RDKit</code> (part I): Reading/Writing, Descriptors, Fingerprints	Lecture06
07	<code>RDKit</code> (part II): Substructure matching, Conformer generation	Lecture07
08	Making a Python package	Lecture08
09	Data Acquisition and Cleaning, Web APIs	Lecture09
10	More packaging; project templates, code testing and coverage.	Lecture10
11	Visualization and analysis of chemical data (clustering)	Lecture11
12	Streamlit	Lecture12
13		

The exercises are where you will learn the most (!), and give you plenty of useful skills for the projects.

- Project-based
- The projects are evaluated along three axis
 - Code (40%)
 - Report (30%)
 - Presentation (30%)
- Groups of 2-3 students
- Multiple people working on the same code, presenting your work.

Project examples from last year

Note: Some of the best grades went to students with limited experience at the beginning of the class.

Welcome to ProtaVision: Delving into Protein Structures and Functions



Welcome to this interactive Jupyter Notebook! 🎉

About This Notebook: This notebook is dedicated to facilitating sequence alignment, a fundamental principle in bioinformatics. Using a custom package, you can input two proteins from a dictionary and efficiently determine the number of matching amino acids, as well as identify the indices of those that do not match. Whether you're delving into bioinformatics for the first time or seeking a streamlined tool for protein analysis, this notebook provides a comprehensive solution.

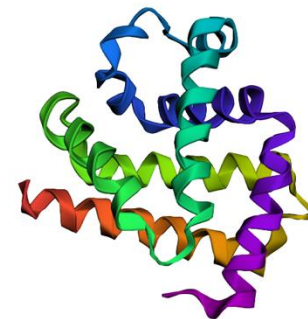
Getting Started: To get started, simply navigate through the sections using the table of contents provided on the ReadMe. Each section is packed with explanations, code snippets, and visualizations to help you grasp concepts effectively.

How to Use: Feel free to experiment with the code! Modify variables, tweak parameters, and run cells to see how different changes affect the outcomes. Don't hesitate to ask questions or seek clarification whenever needed.

Additional Resources: If you're new to Jupyter Notebooks or need a refresher, check out the [Jupyter Notebook Documentation](#) for comprehensive guidance.

Questions? If you have any questions, please contact nichelle.sequeira@epfl.ch or aurelie.masson@epfl.ch

Let's Dive In! Start exploring and discovering insights within this notebook. Happy coding! 🚀



<https://github.com/nichellesequeira/ProtaVision>

<https://github.com/MW21P/rfpred>

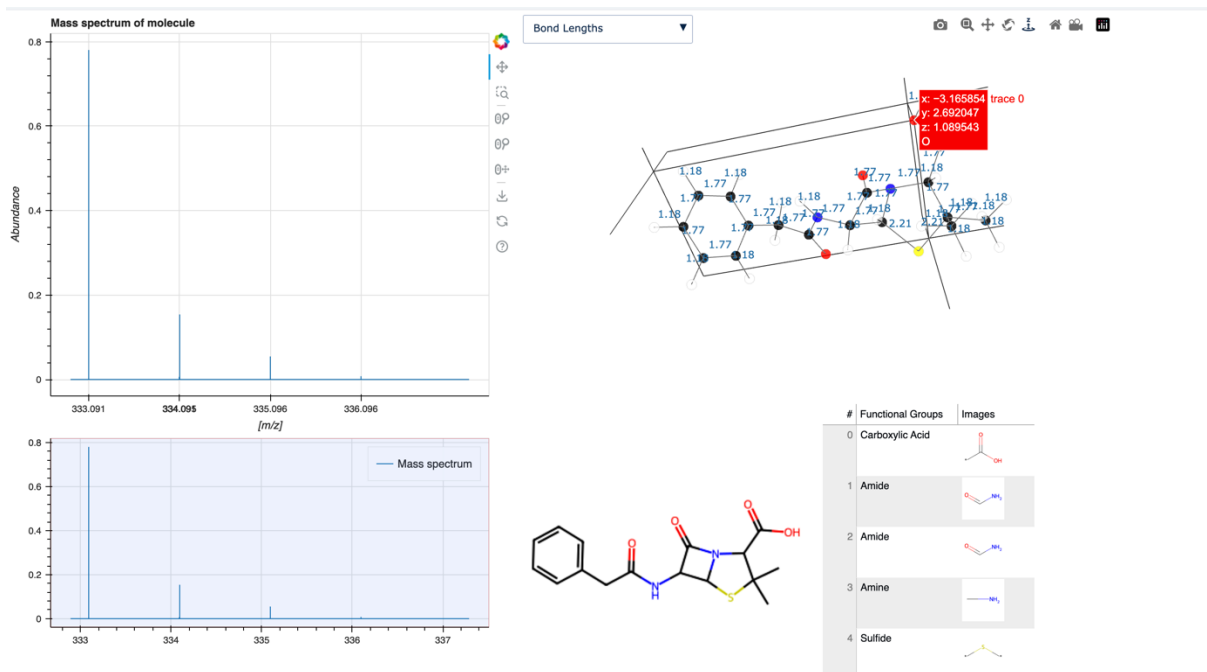


Maintained?	yes	Python	3.10	Contributors	3	License	MIT	EPFL	CH200
-------------	-----	--	------	--------------	---	---------	-----	------	-------

- Python package for applied analytical chemistry focused primarily on mass spectrometry

Project within *practical programming in chemistry* course -- EPFL CH-200

MASSIVEChem, which stands for "Mass Analytical Spectrometry System for Investigation and Visual Extrapolation in Chemistry", is a pip-installable python package developed at EPFL in 2024 focused on, as its name would suggest, analytical chemistry. The aim of this package is to provide the user with functions in order to simulate the mass spectrum of a molecule and to display this spectrum on a graph. The package also provides other features that can facilitate the chemical analysis of a molecule such as a functional group finder and an unsaturation calculator.



MOSER.py (for Molecular Operations and Solutions for Equilibria and Reactions) is a versatile Python package designed for physical chemistry calculations. In honor of Pr. Jacques-Edouard Moser, whose course "Équilibres et Réactivités Chimiques" at EPFL from 2005 to 2022 inspired the creation of this Python package.

Features

Unlocking an abundance of analytical tools, **MOSER.py** serves as an invaluable companion for chemistry students, providing intuitive functionalities for essential calculations and analyses. **MOSER.py** streamlines complex tasks, empowering students to deepen their understanding of physical chemistry concepts and accelerate their learning journey. Among other features, it enables the users to:

1. Balance Chemical Equations

Easily balance chemical equations with a simple function call. **MOSER.py** uses advanced algorithms to ensure accurate balancing of equations.

2. Calculate Molar Mass

Determine the molar mass of chemical compounds effortlessly with **MOSER.py**. Input the chemical formula, and **MOSER.py** will compute the molar mass for you.

3. Determine the Reactional Quotient

Evaluate reaction quotients and kinetics parameters using **MOSER.py**'s intuitive interface.

4. Evaluate a Solution's Concentration and pH

Quickly calculate the concentration of solutions and pH values using **MOSER.py**. Specify the relevant parameters, such as the initial concentration of the solute and the volume of the solution, to obtain accurate results.

5. Simulate Titration Curves

Trace titration curves for acid-base titrations and visualize the pH changes throughout the titration process. Specify the concentrations of the acid or base, as well as the volume of titrant added, to generate detailed titration curves.

```
from Titration import TitrationUI
TitrationUI()
```

Select the type of titration you want to perform.

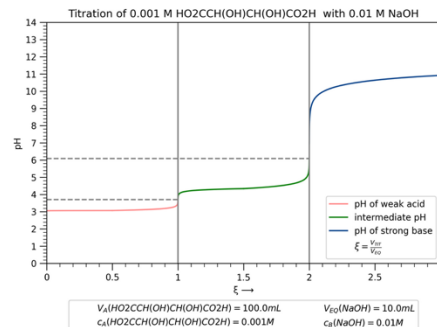
- acid titrated by (strong) base --- 1
- base titrated by (strong) acid --- 2

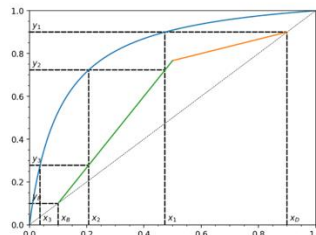
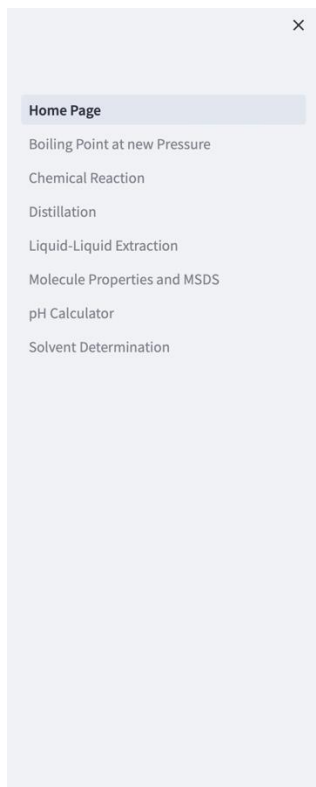
Make your choice (1/2): 1

Please enter your data:

- chemical formula of titrated acid: HO2CCH(OH)CH(OH)CO2H
- (acid) pKa1 [-] = 3.07
- (acid) pKa2 [-] (type '*' if there is none) = 4.34
- initial volume of titrated acid: V_a [L] = 0.100
- chemical formula of titrant base: NaOH
- concentration of titrant base: C_{tit} [mol·L⁻¹] = 0.010
- volume of titrant solution at (first) equivalence point: V_{eq} [L] = 0.010

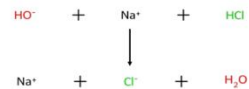
Find hereafter the corresponding titration curve.





Distillation

This app calculates the number of distillation states needed for a binary distillation column.

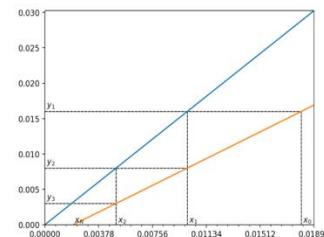


Chemical reaction

This app calculates the stoichiometry of a chemical reaction and the properties of the reactants/products.

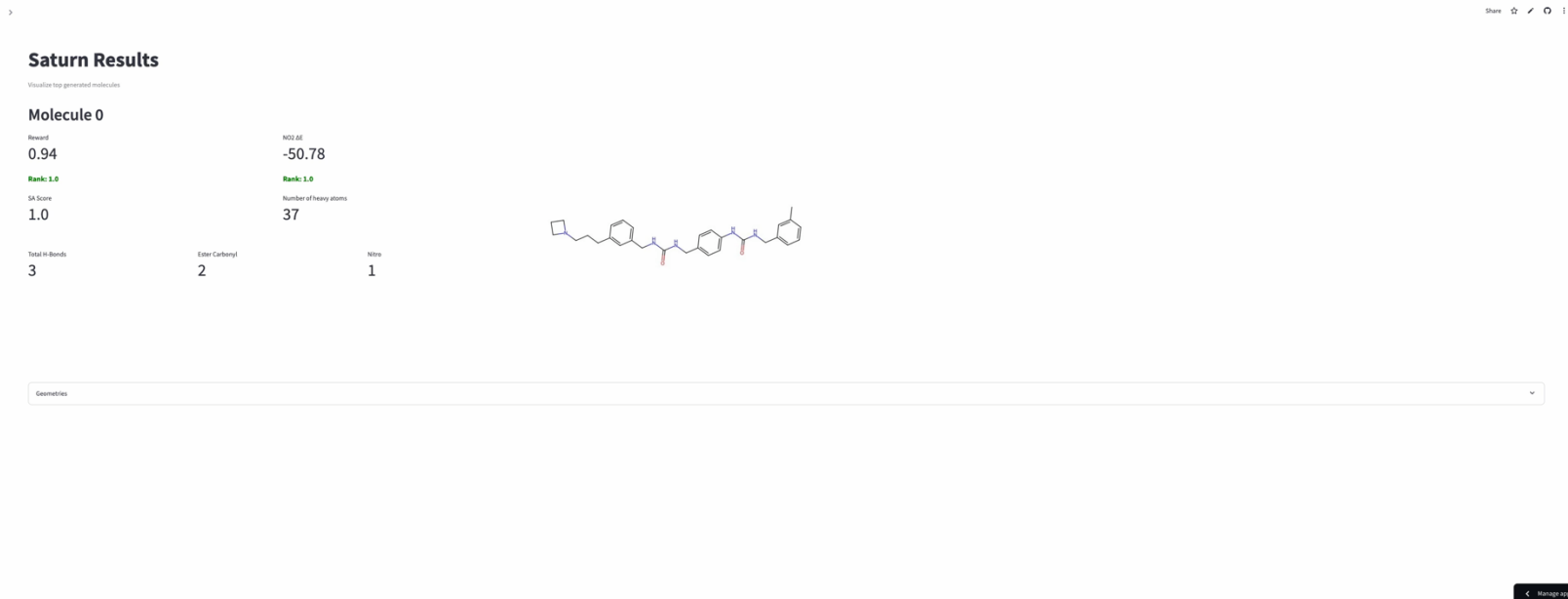


pH Calculator



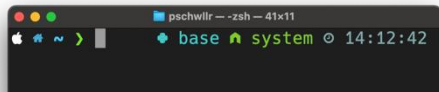
Liquid-Liquid Extraction

A interactive streamlit app with different tools



EPFL The goal for this week – setup your computer

28



Programming language to communicate
with computer system.
Super useful to know a few commands.

Basic command line usage
in bash

General purpose programming language.
Create scripts, programs, webapps, ...

ANACONDA®

Install Python with Conda
(main course programming language)



Visual Studio Code

A better text editor
to handle Python code.

Install VS Code as interactive
development environment

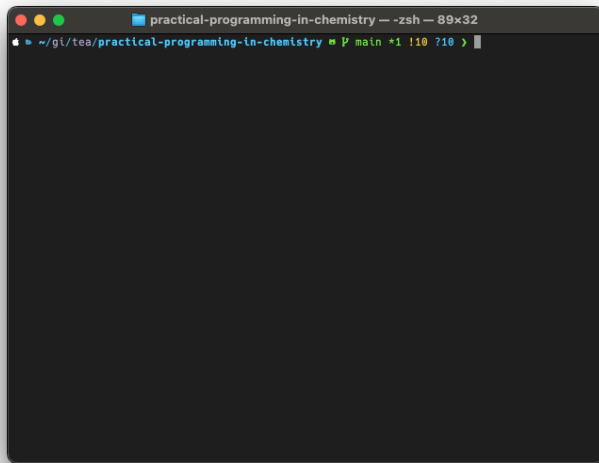
A platform to share, track,
and store code.

GitHub

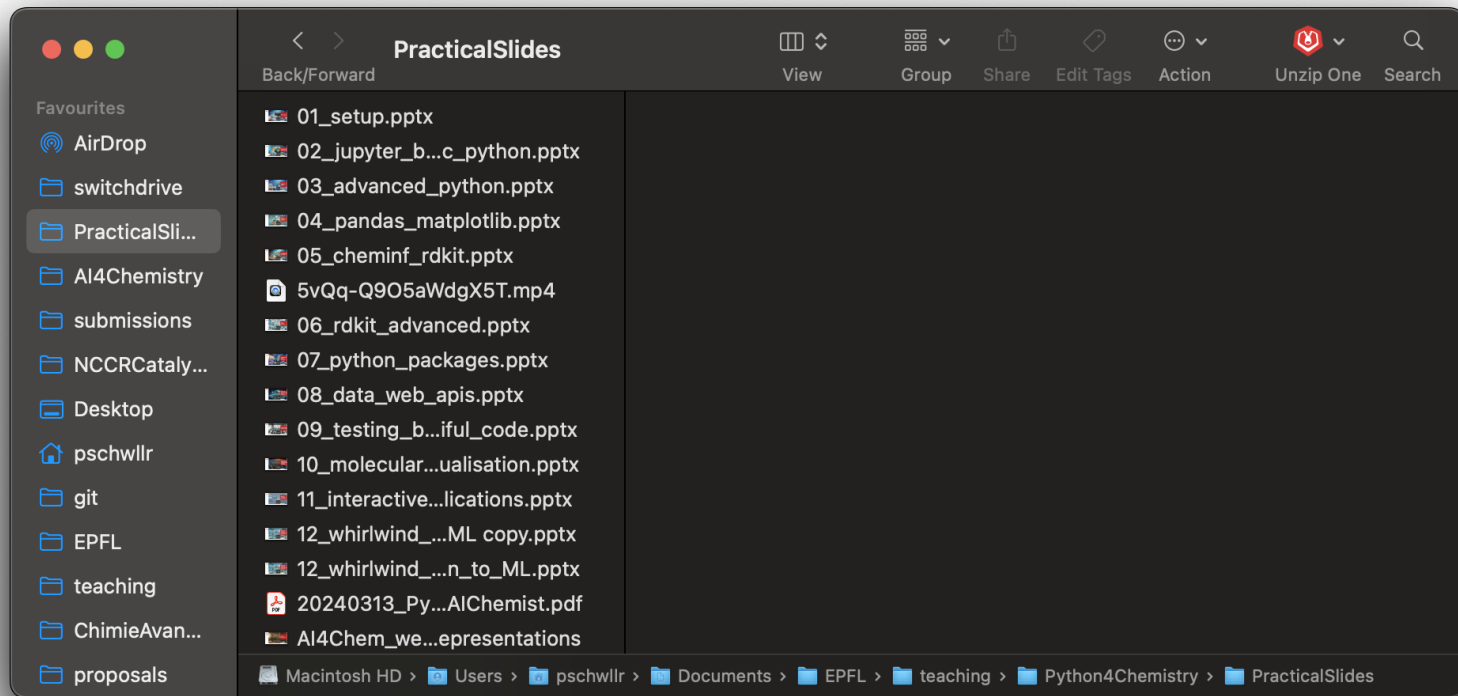
Creating a GitHub account
and getting familiar with basic commands

Using Command line interface (CLI) with bash

- The command line interface (CLI) allows users to **interact with the computer's** operating system or software by **typing text-based commands**.
- MacOS/Linux → Terminal (bash/zsh)
- Windows → **Git Bash**

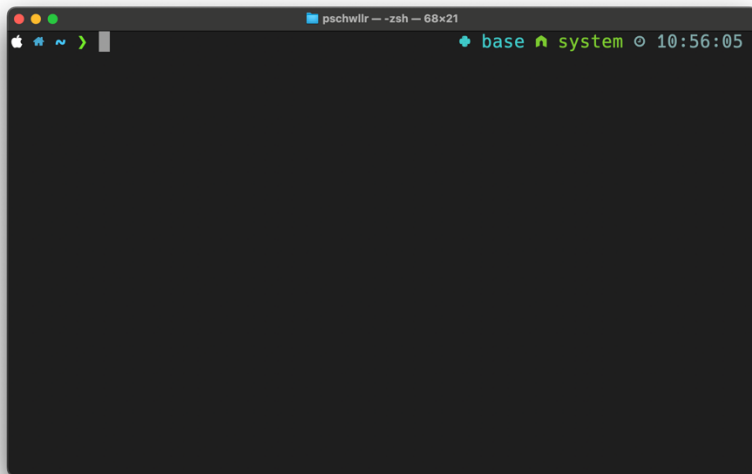


So far, how you navigated through your computer is by clicking through folders



The command line interface gives you the ability to navigate through your computer using commands.

- You can open the interface:
 - Mac: Spotlight (⌘ + Space) and type "**Terminal**"
 - Windows: Start menu and type "**Git Bash**"



Your name The name of your computer

username@computer-name:~\$

After the \$,
is where you can type
commands.

The current folder

- ~ means the home directory
- the starting point when you open a new terminal window

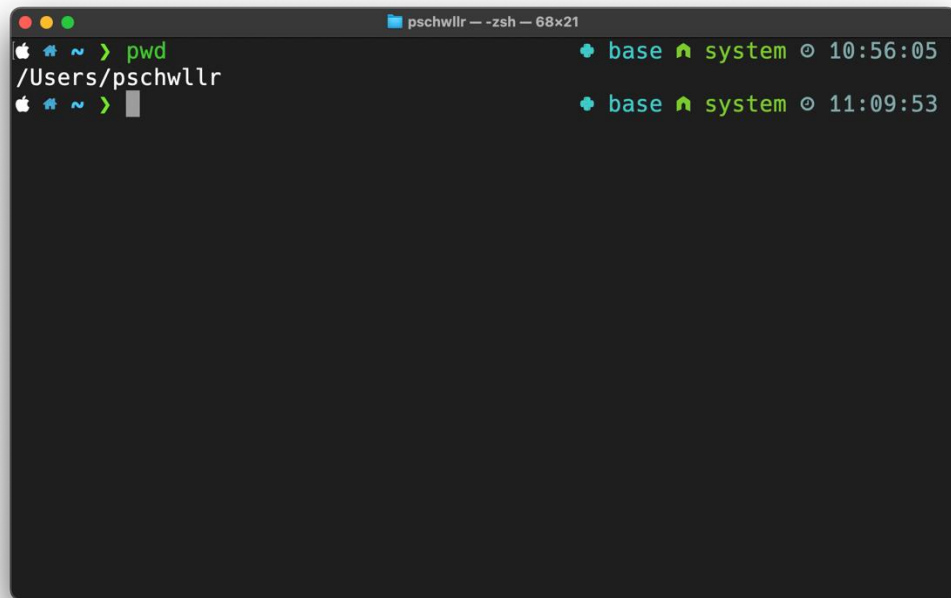
Useful bash commands



bash is the name of the language
for those commands

“pwd” (print current working directory) - where am I currently?

pwd + ENTER → /users/pschwillr, which is my home directory



```
pschwillr -- zsh -- 68x21
base system 10:56:05
/Users/pschwillr
base system 11:09:53
```

A screenshot of a macOS terminal window. The title bar at the top reads "pschwillr -- zsh -- 68x21". The terminal has a dark background with light green text. The first prompt shows the user at the root directory (~) with the command "pwd" entered. The output is "/Users/pschwillr". The second prompt shows the user at the root directory (~) with a cursor waiting for input. On the right side of the terminal, there are two status bars: "base system 10:56:05" and "base system 11:09:53".

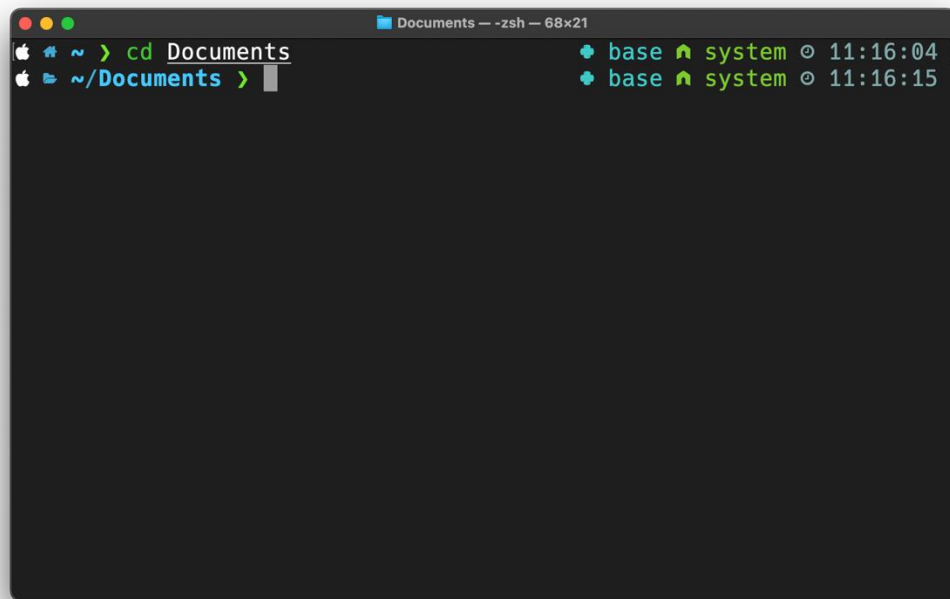
“cd” (change directory) - moving around

```
# Moving around
cd Documents      # Change Directory to Documents
cd ..             # Go up one level
cd ~              # Go to home directory
cd Desktop        # Go to Desktop
cd "My Folder"    # Use quotes for names with spaces
```

“cd” (change directory) - moving around

Moving around

```
cd Documents      # Change Directory to Documents
cd ..             # Go up one level
cd ~              # Go to home directory
cd Desktop        # Go to Desktop
cd "My Folder"    # Use quotes for names with spaces
```

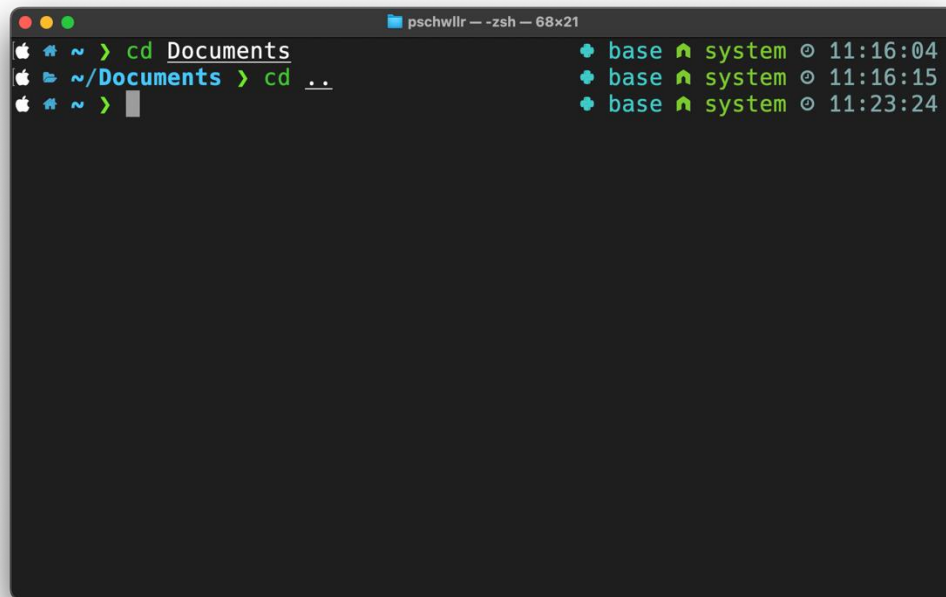
A terminal window titled "Documents — -zsh — 68x21" showing a directory change. The prompt is "~ >". The user enters "cd Documents". The prompt changes to "~/Documents >". On the right side of the terminal, there are two status lines: "base ^ system @ 11:16:04" and "base ^ system @ 11:16:15".

```
Documents — -zsh — 68x21
~ > cd Documents
~/Documents >
base ^ system @ 11:16:04
base ^ system @ 11:16:15
```

“cd” (change directory) - moving around

Moving around

```
cd Documents      # Change Directory to Documents
cd ..             # Go up one level
cd ~              # Go to home directory
cd Desktop        # Go to Desktop
cd "My Folder"    # Use quotes for names with spaces
```



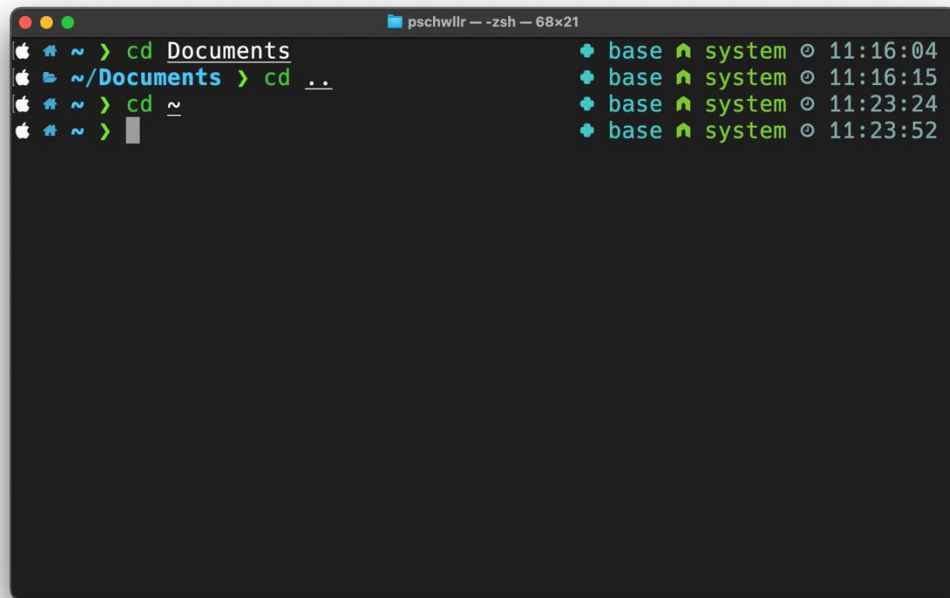
A terminal window titled "pschwillr -- zsh -- 68x21" showing a sequence of commands and system logs. The commands executed are: `cd Documents`, `~/Documents > cd ..`, and `~ >` with a cursor. To the right of the terminal, three system log entries are visible, each starting with a green plus icon, a user icon, and the text "base system" followed by a timestamp.

```
base ~ system @ 11:16:04 |
base ~ system @ 11:16:15 |
base ~ system @ 11:23:24 |
```


“cd” (change directory) - moving around

Moving around

```
cd Documents      # Change Directory to Documents
cd ..             # Go up one level
cd ~              # Go to home directory
cd Desktop        # Go to Desktop
cd "My Folder"    # Use quotes for names with spaces
```



A terminal window titled "pschwillr -- -zsh -- 68x21" showing a sequence of directory navigation commands. The prompt is a tilde (~). The commands entered are: `cd Documents`, `cd ..`, and `cd ~`. To the right of the terminal, a vertical list of system logs shows the user "base" logging in from "system" at various times: 11:16:04, 11:16:15, 11:23:24, and 11:23:52.

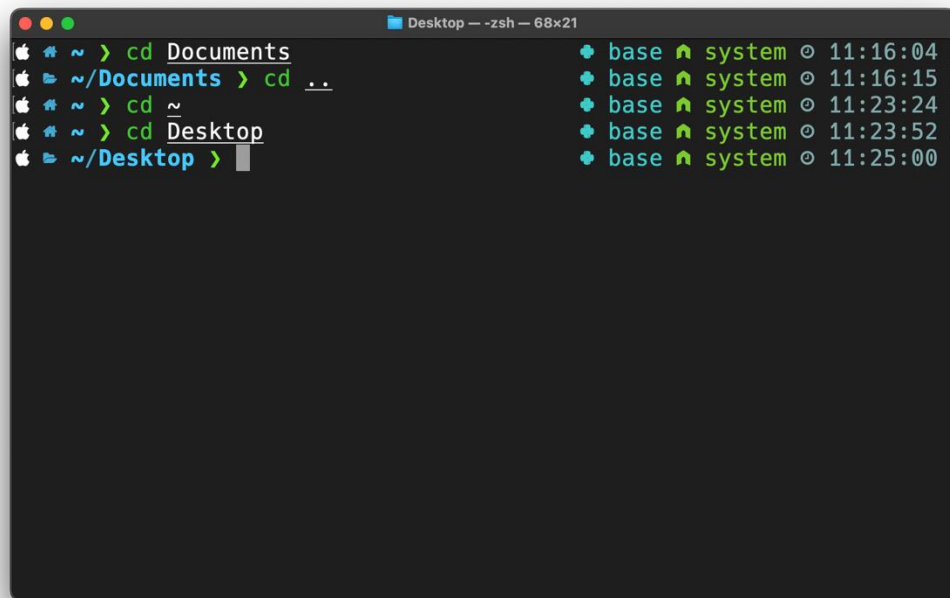
```
pschwillr -- -zsh -- 68x21
~ > cd Documents
~/Documents > cd ..
~ > cd ~
~ > 
```

base system 11:16:04
base system 11:16:15
base system 11:23:24
base system 11:23:52

“cd” (change directory) - moving around

Moving around

```
cd Documents           # Change Directory to Documents
cd ..                  # Go up one level
cd ~                    # Go to home directory
cd Desktop              # Go to Desktop
cd "My Folder"         # Use quotes for names with spaces
```



A terminal window titled "Desktop -- -zsh -- 68x21" showing a sequence of directory navigation commands. The prompt is a blue icon followed by a tilde (~). The commands and their outputs are: `cd Documents`, `cd ..`, `cd ~`, `cd Desktop`, and `~/Desktop`. To the right of the terminal window, there is a vertical list of system logs showing the user "base" on the "system" machine at various times.

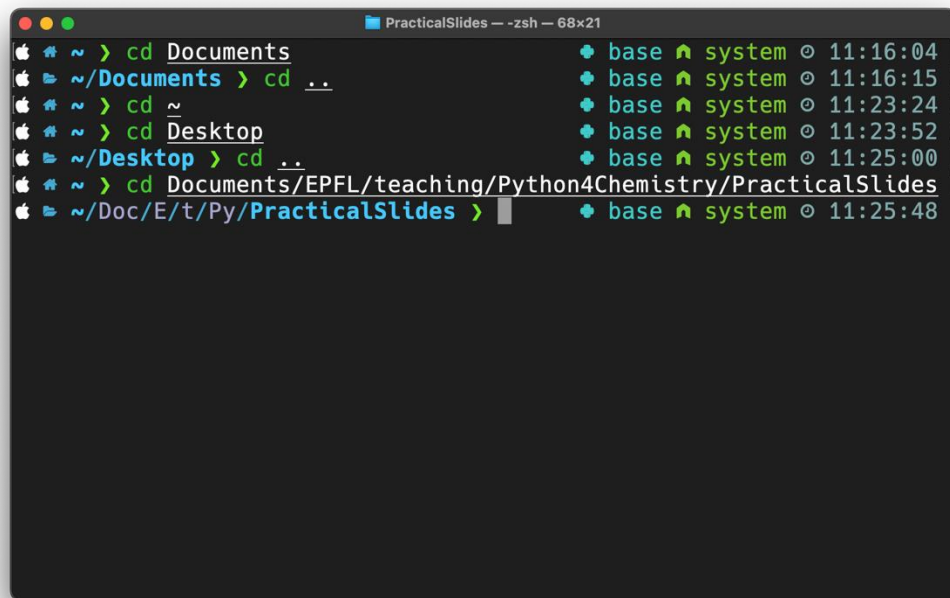
```
Desktop -- -zsh -- 68x21
~ > cd Documents
~/Documents > cd ..
~ > cd ~
~ > cd Desktop
~/Desktop >

base ^ system @ 11:16:04
base ^ system @ 11:16:15
base ^ system @ 11:23:24
base ^ system @ 11:23:52
base ^ system @ 11:25:00
```

"cd" (change directory) - moving around

Moving around

<code>cd Documents</code>	<i># Change Directory to Documents</i>
<code>cd ..</code>	<i># Go up one level</i>
<code>cd ~</code>	<i># Go to home directory</i>
<code>cd Desktop</code>	<i># Go to Desktop</i>
<code>cd "My Folder"</code>	<i># Use quotes for names with spaces</i>



```
PracticalSlides -- -zsh -- 68x21
~ > cd Documents
~/Documents > cd ..
~ > cd ~
~ > cd Desktop
~/Desktop > cd ..
~ > cd Documents/EPFL/teaching/Python4Chemistry/PracticalSlides
~/Doc/E/t/Py/PracticalSlides >
```

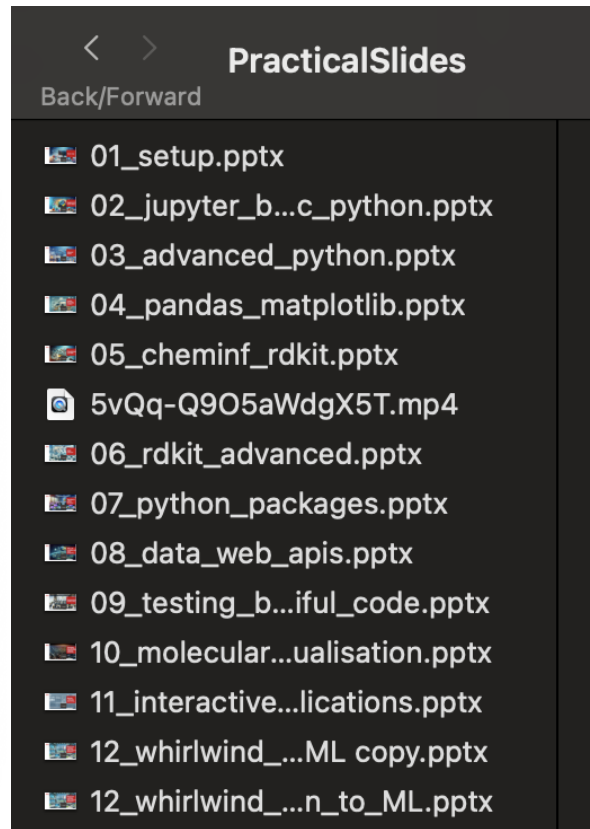
The terminal window also displays system logs on the right side:

```
+ base ^ system @ 11:16:04
+ base ^ system @ 11:16:15
+ base ^ system @ 11:23:24
+ base ^ system @ 11:23:52
+ base ^ system @ 11:25:00
+ base ^ system @ 11:25:48
```

"ls" (list directory)

- what's in the current folder?

```
PracticalSlides -- zsh -- 68x21
~/Documents > cd ..
~ > cd ~
~ > cd Desktop
~/Desktop > cd ..
~ > cd Documents/EPFL/teaching/Python4Chemistry/PracticalSlides
~/Documents/EPFL/teaching/Python4Chemistry/PracticalSlides > ls
01_setup.pptx
02_jupyter_basic_python.pptx
03_advanced_python.pptx
04_pandas_matplotlib.pptx
05_cheminf_rdkit.pptx
06_rdkit_advanced.pptx
07_python_packages.pptx
08_data_web_apis.pptx
09_testing_beautiful_code.pptx
10_molecular_visualisation.pptx
11_interactive_applications.pptx
12_whirlwind_introduction_to_ML copy.pptx
12_whirlwind_introduction_to_ML.pptx
20240313_Python_AIChemist.pdf
5vQq-Q905aWdgX5T.mp4
```

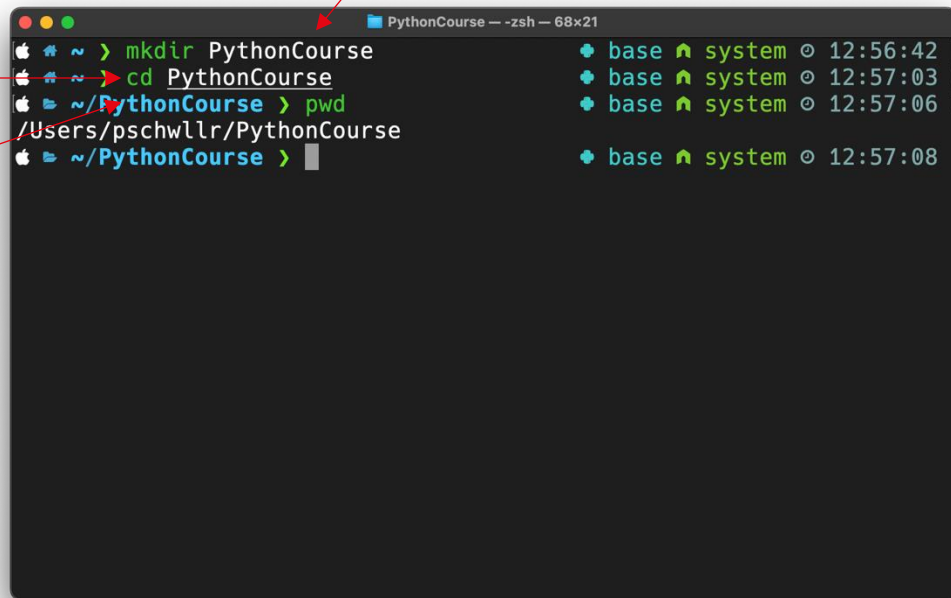


“mkdir” (Make directory)

Create a PythonCourse folder

Navigate to the new folder

Get the path to the current folder.

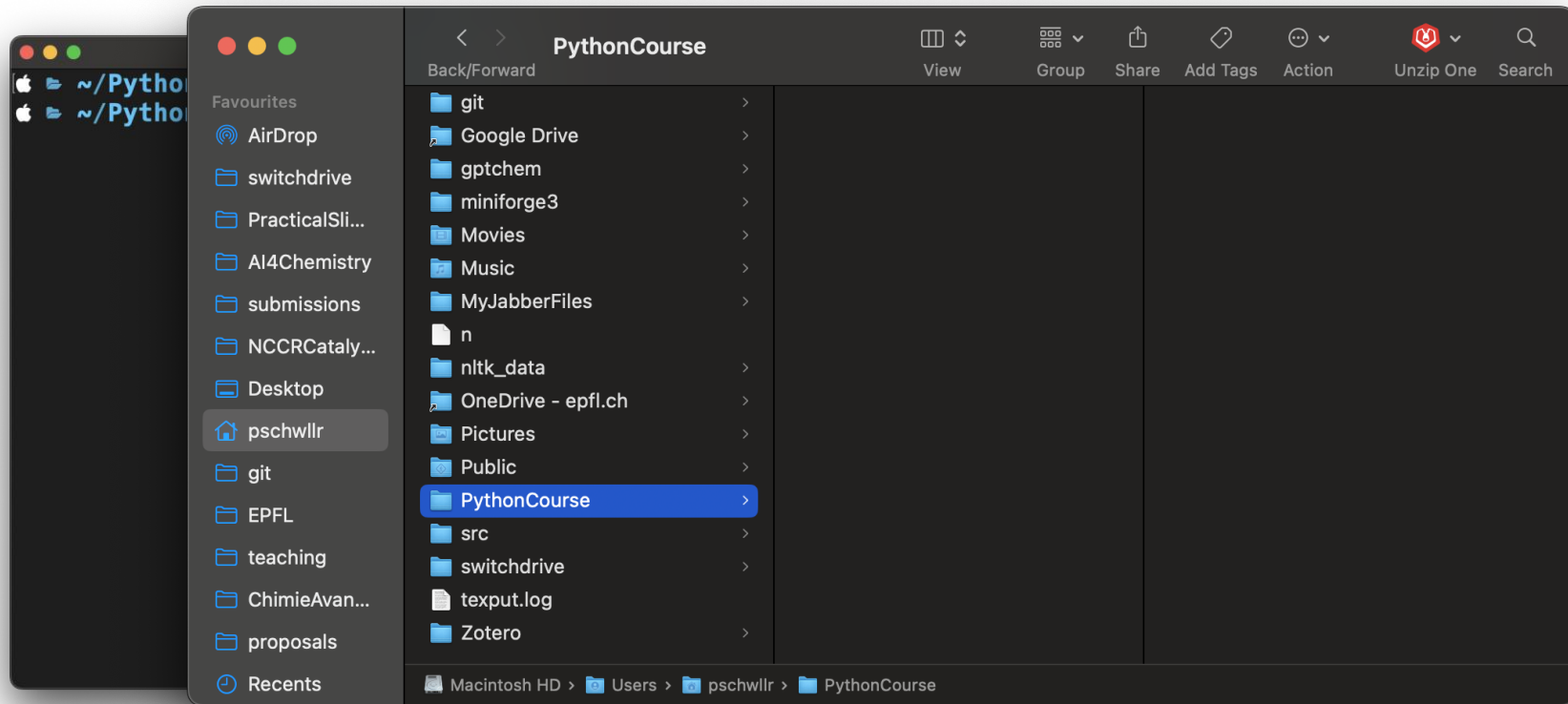


```
PythonCourse -- zsh -- 68x21
base ~ > mkdir PythonCourse
base ~ > cd PythonCourse
base ~/PythonCourse > pwd
/Users/pschwillr/PythonCourse
base ~/PythonCourse >
```

The terminal window shows a sequence of commands: `mkdir PythonCourse` to create the directory, `cd PythonCourse` to navigate into it, and `pwd` to confirm the current path is `/Users/pschwillr/PythonCourse`. Red arrows point from the text annotations to the corresponding commands in the terminal.

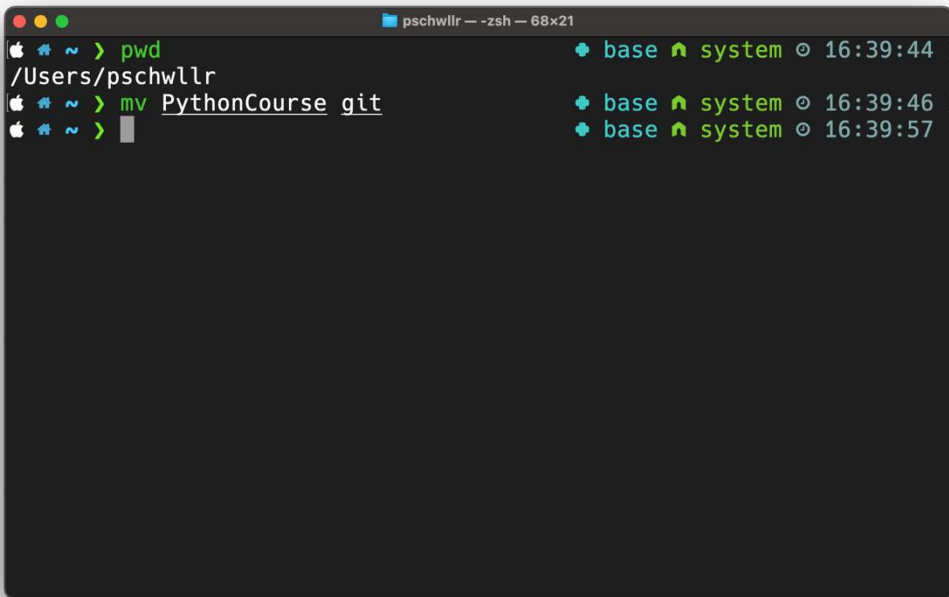
“open ...”

- open either a folder or file



“mv” (move)

- move file or folder / rename



```
pschwillr — -zsh — 68x21
base system 16:39:44
~ > pwd
/Users/pschwillr
base system 16:39:46
~ > mv PythonCourse git
base system 16:39:57
~ > 
```

What are the two things that could happen here?

If the “git” folder exists:
→ It will move “PythonCourse” into “git”

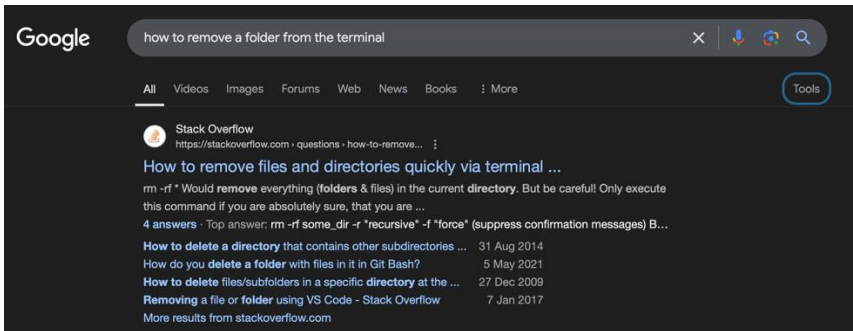
If the “git” folder does not exist:
→ It will rename “PythonCourse” to “git”

“rm” (remove)

- Remove files and directories (be careful)

```
# Creating and removing directories  
mkdir ChemLab      # Make a directory named ChemLab  
rmdir ChemLab      # Remove empty directory  
rm -r ChemLab      # Remove directory and all its contents (be careful!)
```

It's hard to remember all those commands.



how to remove a folder from the terminal

To remove a folder from the terminal, use one of the following commands depending on your needs:

- 1. Remove an empty folder**

```
bash
rm -rf folder_name
```

This only works if the folder is empty.
- 2. Remove a folder and its contents recursively**

```
bash
rm -r folder_name
```

This removes the folder and all files/subdirectories inside it.
- 3. Force delete a folder (dangerous)**

```
bash
rm -rf folder_name
```

- Case sensitivity matters (**Desktop** is different from **desktop**)
- **Spaces in names need quotes** ("My folder")
- The power (and responsibility) of delete commands
- **Tab completion** to avoid typing errors
- **Using up/down arrow keys** to recall previous commands

Bash is even more powerful

Additional commands to work with files

```
# Creating and viewing files
touch data.txt          # Create empty file
cat data.txt            # Display file contents
less data.txt           # View long files (page by page)
head -n 5 data.txt      # Show first 5 lines
tail -n 5 data.txt      # Show last 5 lines

# Copying and moving
cp data.txt backup.txt  # Copy a file
cp -r DataFolder BackupFolder # Copy a directory and contents
mv old.txt new.txt      # Rename/move a file
mv data.txt ../DataFolder/ # Move file to different directory

# Finding things
find . -name "*.txt"    # Find all .txt files in current directory
grep "pH" data.txt      # Search for "pH" in file
grep -r "glucose" .     # Search in all files recursively

# File information
file data.txt           # Show file type
wc -l data.txt          # Count lines in file
du -h data.txt          # Show file size
```

Terminal demo?

```
# Getting help
man ls           # Show manual for ls command
ls --help       # Quick help for ls
history         # Show command history

# System information
date            # Show current date/time
whoami          # Show current user
df -h          # Show disk space usage
top             # Show running processes
```

Bash / Python comparison

- **Bash is a command-line shell language** that's primarily designed to:
 1. Navigate and manipulate files/folders
 2. Run programs and chain them together
 3. Automate system tasks
- **Python is a general-purpose programming language** that's great for:
 1. Data analysis and visualization
 2. Scientific computing
 3. Writing complex algorithms
 4. Creating applications

```
python

# Python script to do the actual analysis
import pandas as pd
import numpy as np

def analyze_spectrum(data):
    # Complex mathematical operations
    # Statistical analysis
    # Machine learning
    # Plotting results
```

```
bash

# Bash script to prepare data for analysis
for file in *.raw; do
    # Convert raw instrument files to CSV
    converter "$file" "${file%.raw}.csv"
done

# Now run the Python analysis
python analyze_spectrum.py
```

Use Python

To run this Python script.

EPFL The goal for this week – setup your computer

53



Programming language to communicate with computer system.
Super useful to know a few commands.

Basic command line usage
in bash

General purpose programming language.
Create scripts, programs, webapps, ...

ANACONDA®

Install Python with Conda
(main course programming language)



Visual Studio Code

A better text editor
to handle Python code.

Install VS Code as interactive
development environment

A platform to share, track,
and store code.

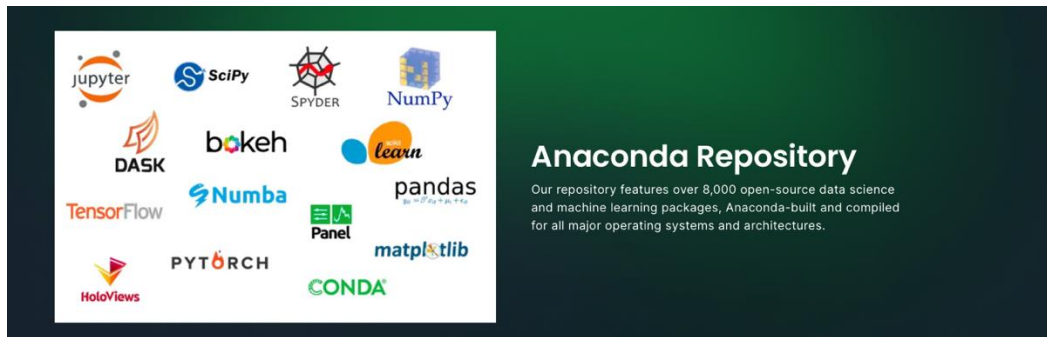
GitHub

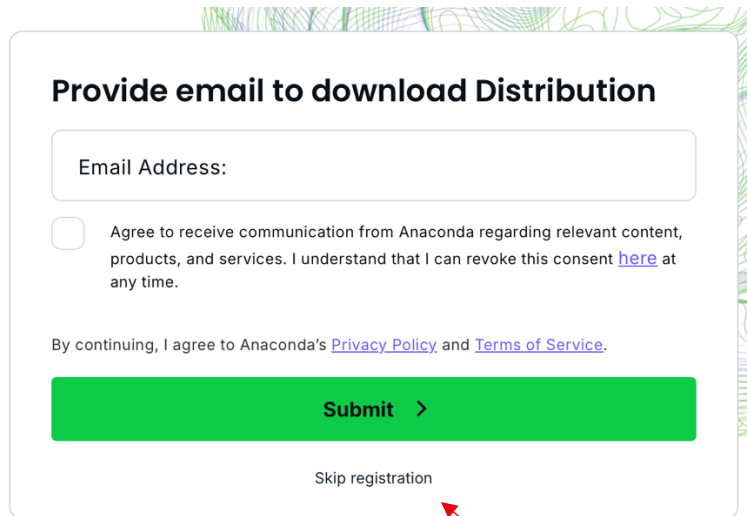
Creating a GitHub account
and getting familiar with basic commands

Anaconda – Python distribution

- Designed for scientific computing, data science, and machine learning.
- Gives you access to thousands of pre-packaged modules
- Simplified **package management system**
- **Environment management**

You can download it from: <https://www.anaconda.com/download>





Provide email to download Distribution

Email Address:

☐ Agree to receive communication from Anaconda regarding relevant content, products, and services. I understand that I can revoke this consent [here](#) at any time.

By continuing, I agree to Anaconda's [Privacy Policy](#) and [Terms of Service](#).

Submit >

[Skip registration](#)

Skip registration

Anaconda Installers

[Download for Mac](#)

Windows

Python 3.12

📄 64-Bit Graphical Installer (912.3M)



Mac

Python 3.12

📄 64-Bit (Apple silicon) Graphical Installer (704.7M)

📄 64-Bit (Apple silicon) Command Line Installer (707.3M)

📄 64-Bit (Intel chip) Graphical Installer (734.7M)

📄 64-Bit (Intel chip) Command Line Installer (731.2M)



Linux

Python 3.12

📄 64-Bit (x86) Installer (1007.9M)

📄 64-Bit (AWS Graviton2 / ARM64) Installer (800.6M)

📄 64-bit (Linux on IBM Z & LinuxONE) Installer (425.8M)

*Note: You will have time to do that in the exercise session.
If you are unsure, which one, we are happy to help.
Wait until the exercises!*

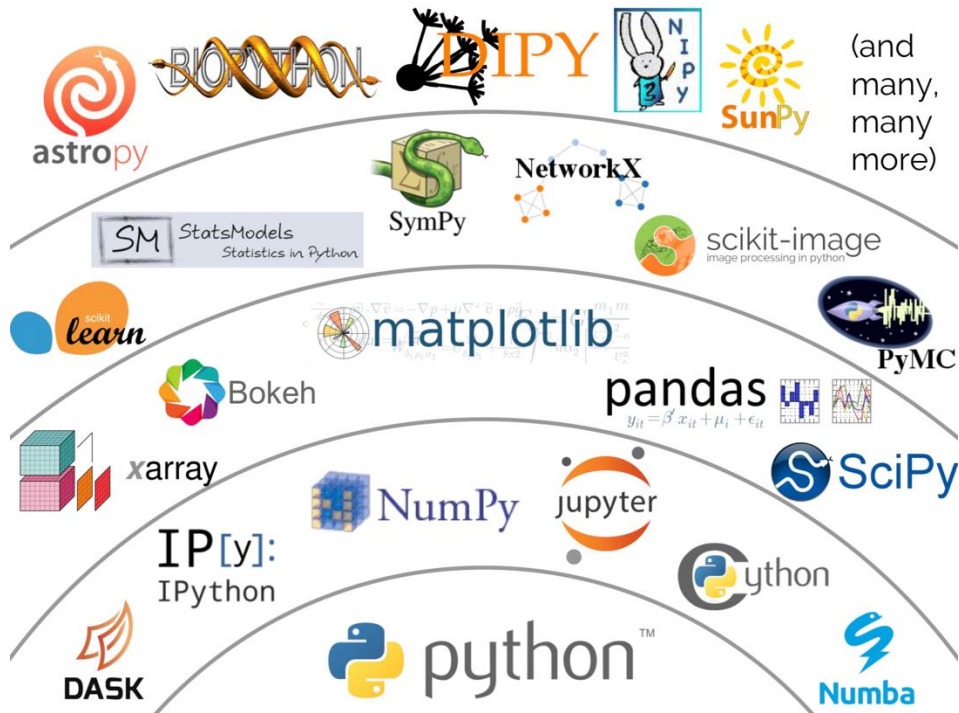


Image: <https://blueskyproject.io>

EPFL The goal for this week – setup your computer

62



Programming language to communicate with computer system.
Super useful to know a few commands.

Basic command line usage
in bash

General purpose programming language.
Create scripts, programs, webapps, ...

ANACONDA®

Install Python with Conda
(main course programming language)



Visual Studio Code

A better text editor
to handle Python code.

Install VS Code as interactive
development environment

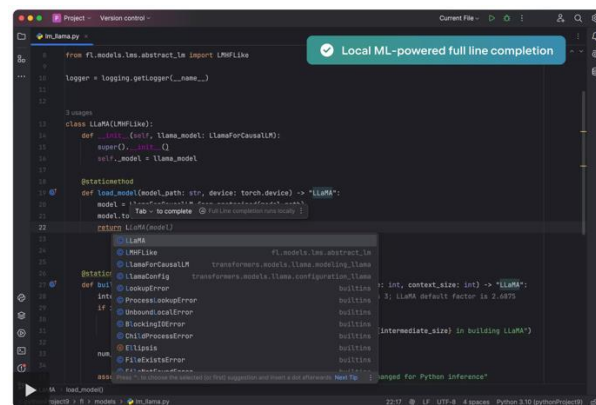
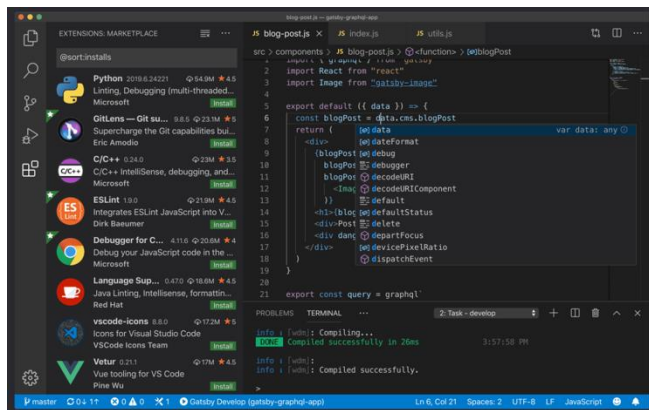
A platform to share, track,
and store code.

GitHub

Creating a GitHub account
and getting familiar with basic commands

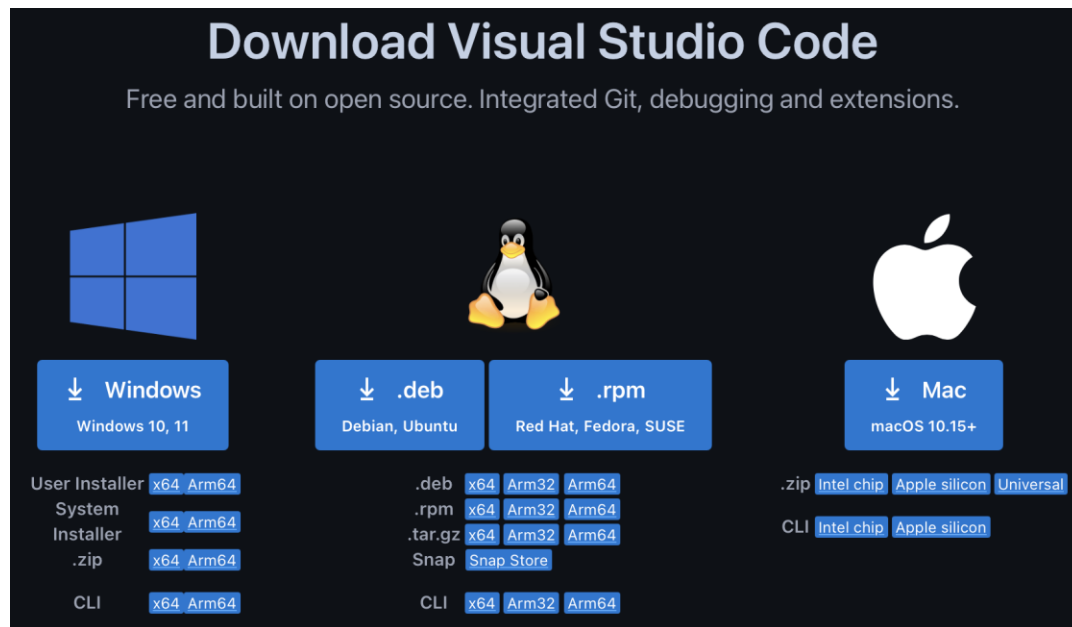
Where to write code? In an interactive development environment (IDE)

- You could use a **text editor**.
- But IDEs provide useful tools for **coding, error checking, autocompletion, etc...**
- Popular examples are **VS Code**, **PyCharm**, and **Jupyter lab**



Here we will use VS Code (you most likely already have it from last year)

- <https://code.visualstudio.com/download>



The screenshot shows the Visual Studio Code download page with a dark blue background. At the top, it says "Download Visual Studio Code" in white, followed by "Free and built on open source. Integrated Git, debugging and extensions." Below this, there are three main sections for operating systems: Windows, Linux, and Mac. Each section has a download button and a list of available installers or packages.

Download Visual Studio Code
Free and built on open source. Integrated Git, debugging and extensions.

Windows
Windows 10, 11

Linux
Debian, Ubuntu: .deb
Red Hat, Fedora, SUSE: .rpm

Mac
macOS 10.15+

Available Installers/Packages:

Platform	Installer/Package	Architectures
Windows	User Installer	x64, Arm64
	System Installer	x64, Arm64
	.zip	x64, Arm64
	CLI	x64, Arm64
Linux	.deb	x64, Arm32, Arm64
	.rpm	x64, Arm32, Arm64
	.tar.gz	x64, Arm32, Arm64
	Snap	Snap Store
	CLI	x64, Arm32, Arm64
Mac	.zip	Intel chip, Apple silicon, Universal
	CLI	Intel chip, Apple silicon

EPFL The goal for this week – setup your computer

65



Programming language to communicate with computer system.
Super useful to know a few commands.

Basic command line usage
in bash



Visual Studio Code

A better text editor
to handle Python code.

Install VS Code as interactive
development environment

General purpose programming language.
Create scripts, programs, webapps, ...

ANACONDA®

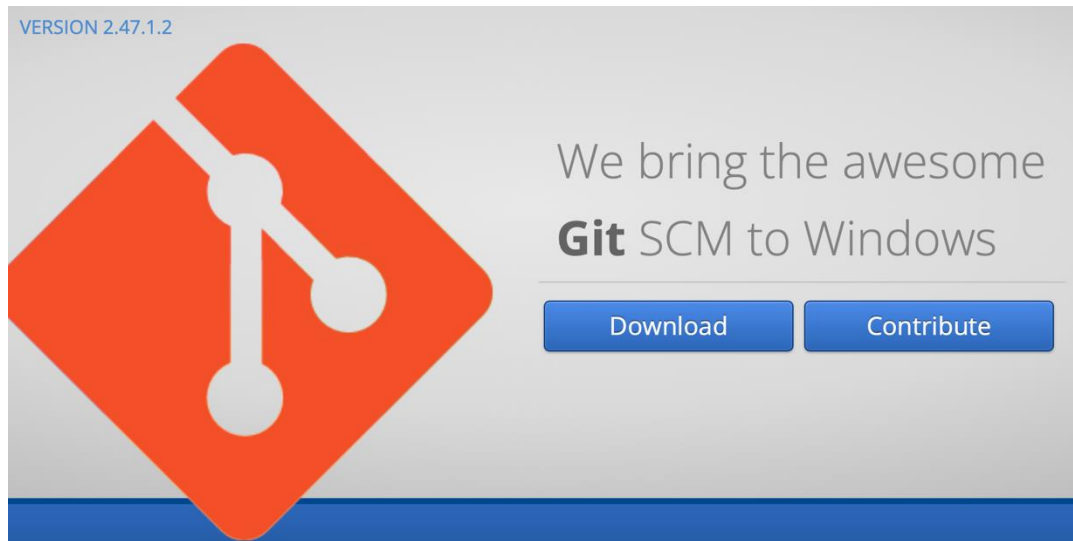
Install Python with Conda
(main course programming language)

A platform to share, track,
and store code.

GitHub

Creating a GitHub account
and getting familiar with basic commands

- <https://gitforwindows.org>
- Download and install it (if you have not already done so)
- Added benefit
→ Git bash
- Mac users should have git natively installed.



GitHub – the platform we will use to share the exercises

Git is a Version Control System (VCS) ideal for

- tracking changes,
- collaborating,
- sharing in coding projects,

allowing users to revert to older versions and work together seamlessly.

GitHub (based on Git) offers an online platform with an easy-to-use interface. Almost a social platform for coding.


<https://github.com/schwallergroup/practical-programming-in-chemistry-exercises>

Build and ship software on a single, collaborative platform

Join the world's most widely adopted AI-powered developer platform.

[Sign up for GitHub](#)[Try GitHub Copilot](#)

- <https://education.github.com/pack>
- Give you free access to unlimited private repositories (and other things people pay for usually)




About GitHub Pages
Websites for you and your projects. Hosted directly from your GitHub repository. Just edit, push, and your changes are live.

Offer
Get one site per GitHub account and organization, and unlimited project sites.

Tags
Developer tools

Get help at [GitHub Pages support](#)




LaunchPad

Profile README

Say 'Hello world' with a profile README that lets you introduce yourself to the GitHub community. You decide what information to include in your profile README, so you have full control over how you present yourself on GitHub. This experience walks you through a quick tutorial to create your profile README.

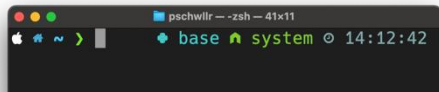
Offers in this bundle

 GitHub

[Learn more about Profile README >](#)

EPFL The goal for this week – setup your computer

86



Getting to know
the command line and bash.

Basic command line usage
in bash

Downloading Anaconda

ANACONDA®

Install Python with Conda
(main course programming language)



Visual Studio Code

Installing VS Code

Install VS Code as interactive
development environment

Making a GitHub account

GitHub

Creating a GitHub account
and getting familiar with basic commands

Most of the exercises, are just making sure, you are properly set up.

And then, there is a game to learn bash.

Download and unpack

If you haven't downloaded Bashcrawl yet, go to <https://gitlab.com/slackermmedia/bashcrawl/-/releases> and download the `Source code (zip)` file.

Once the ZIP file has been downloaded to your computer, unzip it as usual.

Now it's time to try it out on your computer.

<https://gitlab.com/slackermmedia/bashcrawl>

stable-2024.02.09

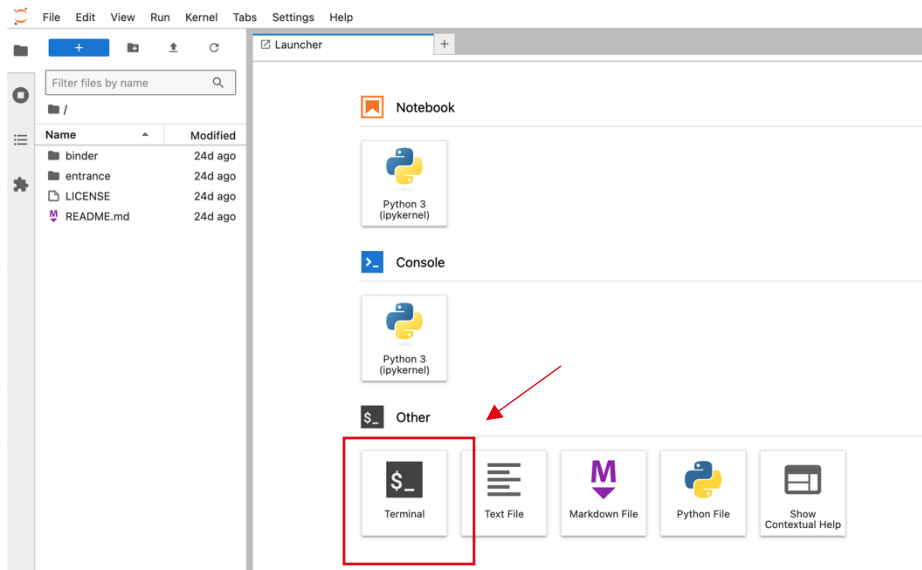
▼ **Assets** 4

 [Source code \(zip\)](#) 

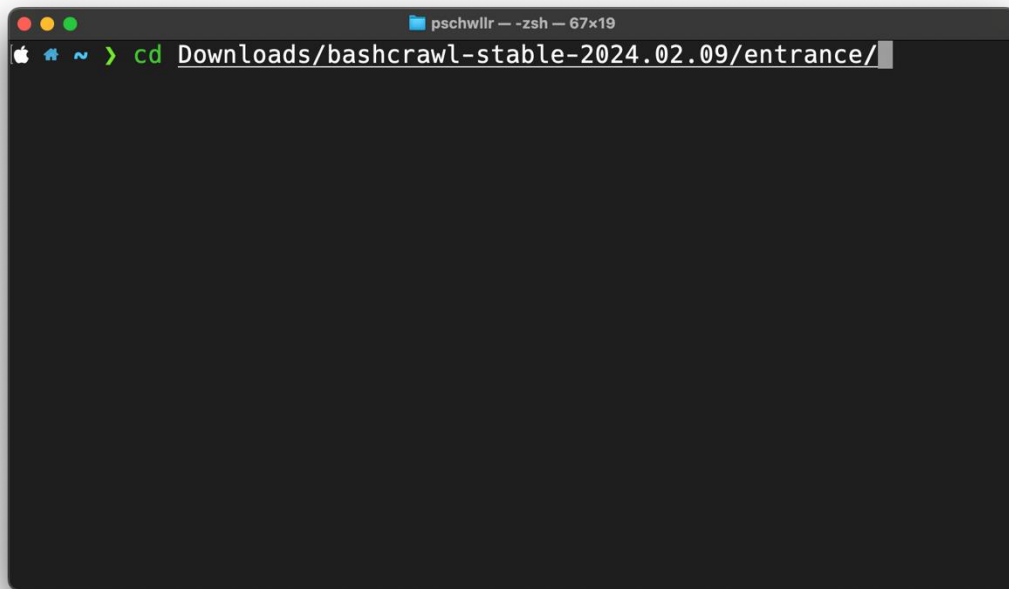
<https://gitlab.com/slackermmedia/bashcrawl/-/archive/stable-2024.02.09/bashcrawl-stable-2024.02.09.zip>

Alternative launch it online with binder (just click on the link and then, the terminal)

- <https://mybinder.org/v2/gl/nthiery%2Fbashcrawl/HEAD>



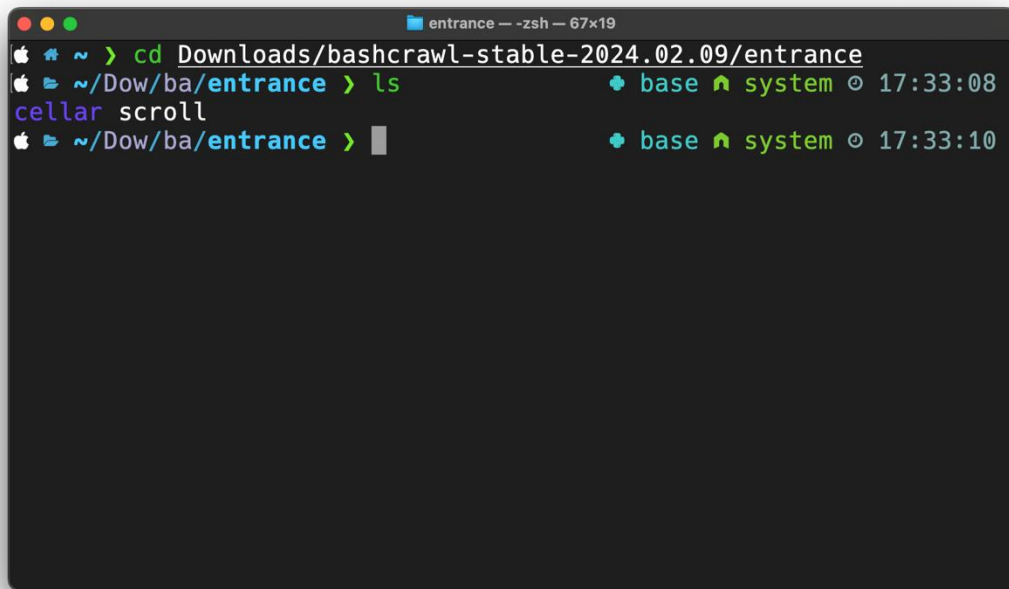
After the download, you can navigate to the entrance with `cd`



A terminal window titled "pschwillr -- zsh -- 67x19" is shown. The prompt is a green "\$" followed by "cd". The command "Downloads/bashcrawl-stable-2024.02.09/entrance/" is entered and highlighted with a white cursor. The terminal background is dark gray.

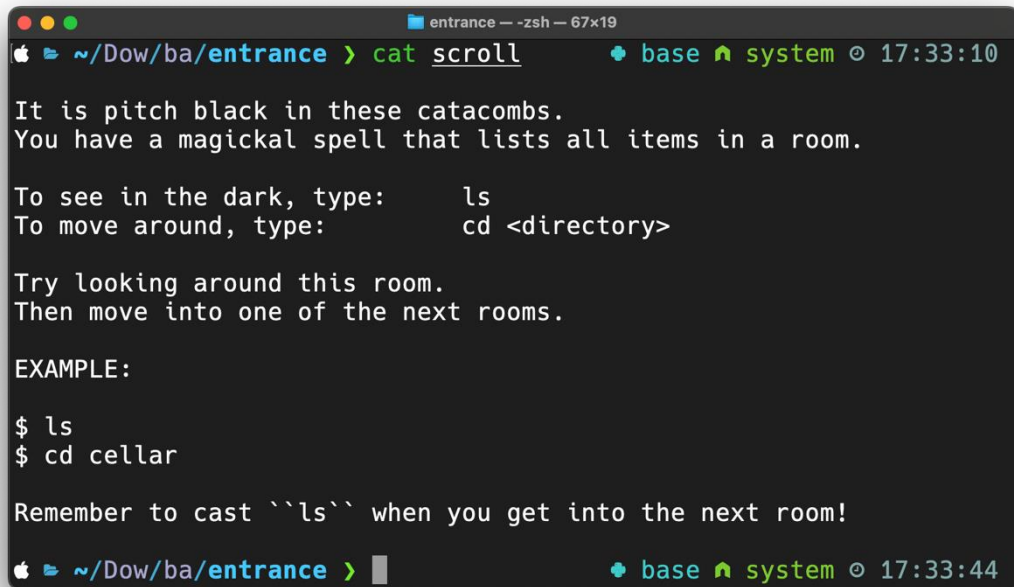
```
pschwillr -- zsh -- 67x19  
$ cd Downloads/bashcrawl-stable-2024.02.09/entrance/
```


ls will list all files in the folder



```
entrance -- zsh -- 67x19
~ > cd Downloads/bashcrawl-stable-2024.02.09/entrance
~/Dow/ba/entrance > ls
cellar scroll
~/Dow/ba/entrance >
```

Every room has a text file called “scroll”



```
entrance --zsh-- 67x19
~/Dow/ba/entrance > cat scroll

It is pitch black in these catacombs.
You have a magickal spell that lists all items in a room.

To see in the dark, type:      ls
To move around, type:         cd <directory>

Try looking around this room.
Then move into one of the next rooms.

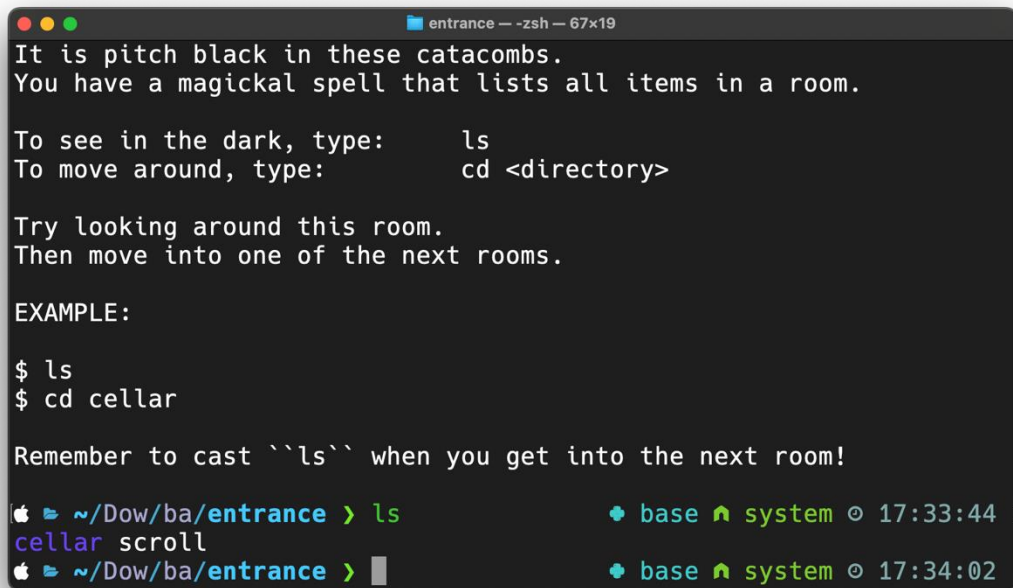
EXAMPLE:

$ ls
$ cd cellar

Remember to cast ``ls`` when you get into the next room!

~/Dow/ba/entrance > |
```

With “ls” you see that there is a room (or folder) called cellar.



A terminal window titled "entrance -- zsh -- 67x19" displays the following text:

```
It is pitch black in these catacombs.  
You have a magickal spell that lists all items in a room.  
  
To see in the dark, type:      ls  
To move around, type:        cd <directory>  
  
Try looking around this room.  
Then move into one of the next rooms.  
  
EXAMPLE:  
  
$ ls  
$ cd cellar  
  
Remember to cast ``ls`` when you get into the next room!
```

At the bottom, there are two terminal prompts showing the user's location and the time:

```
🍏 ~/Dow/ba/entrance > ls                               🍏 base 🏠 system 🕒 17:33:44  
cellar scroll  
🍏 ~/Dow/ba/entrance >                                   🍏 base 🏠 system 🕒 17:34:02
```

“cd cellar”, then “cat scroll”, and you have the next instructions

```
cellar -- zsh -- 67x25
base ^ system 17:34:02
base ^ system 17:34:44

# Illusions are strong here.
# It is difficult to tell what is a doorway and what is an object.
#
# The magic spell you use to look can be augmented.
#
# From now on, cast your spell like this:
#
# ls -F
#
# Directories (the rooms of these catacombs) end with a / symbol.
#
# Encounters (programs) end with a * symbol.
#
# You can avoid having to type `ls -F` every time by running the
# following:
#
# alias ls='ls -F'
#
# This is known as a shell or command alias. With this alias,
# typing simply ls by itself will run ls -F. Try it out!
base ^ system 17:34:48
```

```
armoury -- zsh -- 67x25
# can come back to this treasure later!

base ^ system 17:38:21
base ^ system 17:38:26
base ^ system 17:38:26

~/Dow/ba/e/cellar > cd armoury
~/Dow/ba/e/c/armoury > ls
chamber/      potion*      scroll      treasure*
~/Dow/ba/e/c/armoury > ./potion
You have found a potion bottle of swirling
green liquid. Do you want to drink it? y/n

y
The taste of a rustic green plant fills your mouth. It
warms and strengthens you.

Create a variable for your health points (HP). You have
15HP:

export HP=15

You can check your health at any time:

echo $HP

base ^ system 17:38:47
```

It is fun to explore that cellar, and run the bash commands that are suggested.

Now, there will be a short break of 5 minutes, and then, the TA will introduce the exercises.